

# PERCEPTUAL AND ACTION ROUTINES IN DIAGRAMMATIC REASONING FOR ENTITY-REIDENTIFICATION

Bonny Banerjee and B. Chandrasekaran\*  
Department of Computer Science & Engineering  
The Ohio State University, Columbus, OH 43210  
{Banerjee, Chandra}@cse.ohio-state.edu

## ABSTRACT

The Objective Force requirements of responsiveness, agility and versatility call for digitized graphical decision support interfaces that automate or otherwise help in various reasoning tasks, including reasoning with visual and diagrammatic representations that are ubiquitous in Army situation understanding, planning and plan monitoring. In earlier papers, we described a diagrammatic reasoning architecture, and demonstrated the approach for instances of *maneuver recognition* and information fusion for *entity-reidentification* problems. The current paper characterizes the computational properties of the core perception and path-finding algorithms that are an important part of the technology's application for a class of fusion problems. This analysis is important since the practicality of automating diagrammatic reasoning for Army applications depends on developing algorithms with manageable complexity.

## 1. INTRODUCTION

Reasoning with visual representations consisting of terrain maps with an overlay of diagrammatic elements is ubiquitous in Army situation understanding, planning and plan monitoring. Diagrams are overlaid on top of terrain maps, and they represent information using a combination of iconic and spatially veridical elements. The overall problem solving process is a sequence of steps each of which is one of three types: *perception* on the diagram, in which information about the spatial properties of or relations between diagrammatic objects is obtained; *inference*, making use of currently available symbolic information including information obtained by perception; and *actions* on the diagram, in which diagrammatic elements are added, deleted or modified to satisfy certain constraints, such as "find a path that goes from point A to point B, while avoiding region C." Automating or semi-automating such reasoning tasks is essential if the ambitious goals of Army Transformation based on information dominance are to be achieved.

We have been experimenting with an architecture [Chandrasekaran, et al, 2002; Chandrasekaran, et al, 2004] for automated reasoning with diagrams and applied it to example problems in maneuver recognition and information fusion for entity reidentification. In this paper, we present the algorithms and their computational



Fig 1. A diagram in ASAS.

complexities for the set of perceptual and action routines that we have found useful in spatial reasoning tasks involved in certain types of information fusion. This analysis is important since the practicality of automating diagrammatic reasoning for Army applications depends on developing algorithms with manageable complexity.

## 2. DIAGRAMS IN A FUSION EXAMPLE

The entity reidentification problem arises in systems such as U.S. Army's All-Source Analysis System (ASAS). The prototypical task can be characterized as follows. A report is received about the sighting of an entity of interest, along with the time, location and partial identity information, such as that it was a tank, or tank of a given type, etc. The task is to decide if the newly sighted object is one of the objects in the database, sighted and identified earlier, or a new object. The overall reasoning process is modeled as abductive inference [Josephson & Josephson, 1996]. The reasoning system has a number of diagrammatic subtasks. Fig 1 illustrates some of them. The three regions are marked as no-go areas for the vehicle type of interest. The newly sighted vehicle is the small circle at the bottom right of the figure, and the database has identified two previously sighted and identified vehicles, the two small circles at bottom left and top of the figure, as being potentially the same as the newly sighted vehicle. The problem solver asked the diagrammatic reasoner to identify a possible path from the new sighting location to the vehicle at the top. The action component of the diagrammatic reasoner found a path

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE <b>00 DEC 2004</b>		2. REPORT TYPE <b>N/A</b>		3. DATES COVERED <b>-</b>	
4. TITLE AND SUBTITLE <b>Perceptual And Action Routines In Diagrammatic Reasoning For Entity-Reidentification</b>				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>Department of Computer Science &amp; Engineering The Ohio State University, Columbus, OH 43210</b>				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release, distribution unlimited</b>					
13. SUPPLEMENTARY NOTES <b>See also ADM001736, Proceedings for the Army Science Conference (24th) Held on 29 November - 2 December 2005 in Orlando, Florida. , The original document contains color images.</b>					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT <b>UU</b>	18. NUMBER OF PAGES <b>8</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

between the two no-go regions. The two elliptical regions marked Sensor1 and Sensor2 are known sensor fields. The problem solver wants to know if the path intersects any of the sensor fields. The diagrammatic reasoner replies, as we would, that the path intersects Sensor1, but not Sensor2. Then the problem solver asks (not shown in Fig 1) if the path could be modified so as to avoid the field Sensor1 but still go between the two no-go regions. The relevant path modification algorithm finds that this is not possible. Another common example of perception is that of *emergent* objects. For instance, when two curves intersect, a new point object, the intersection point, is created. This may be of significance in some domain, e.g., that point might be an intersection between two routes, providing alternatives for path planning. Posing a sequence of such questions to, and making use of the answers from, the diagrammatic component, the problem solver eventually decides that the new sighting could not correspond to the one at the top.

### 3. DIAGRAMMATIC REPRESENTATION

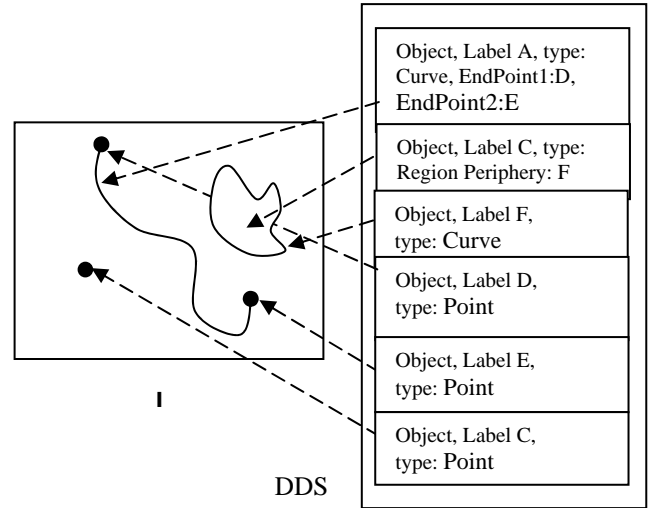
It is important to distinguish between a diagram and a general image. A diagram is first and foremost a *representation*, i.e., it is not an image of a natural object, but one whose objects are intended to represent information to support some reasoning task. A diagram differs from linguistic-symbolic representations in that the spatial properties of and relations between diagrammatic objects may be used to represent information in the domain.

A Diagram is a pair ( $I$ , DDS) where  $I$  is the image, defined as a specification – implicit or explicit – of intensity values for points in the relevant regions of 2D space, and DDS is the *Diagram Data Structure*, which is a list of labels for the diagrammatic objects in the image; associated with each object label is a specification of the subset of  $I$  that corresponds to the object. A diagrammatic object can be one of three types: *point*, *curve*, and *region*. Point objects only have location (no spatial extent), curve objects only have axial specification (no thickness), and region objects have location and spatial extent. The labels are internal to DDS. External labels such as A, B, etc. in Fig 2 are additional features of the objects that may be associated with them.  $I$  is any description from which a specification of intensity values for the relevant points in the 2D space can be obtained. In our work we represent curve objects by a sequence of points or line segments, and regions by the perimeter closed curves, also as line segments.

Diagrams are constructed by placing diagrammatic objects on a 2D surface in specific configurations to represent specific information. DDS will initially consist of the labels of objects so placed and their spatial specifications.

A perceptual routine<sup>1</sup> (PR) takes a specified number of diagrammatic objects in the diagram and returns a perception, which may be an object, a property or a relation. An action routine (AR) creates, deletes or modifies objects in the diagram so as to satisfy given constraints, some of which may be perceptual.

There are no finite sets of these routines that suffice for all of diagrammatic reasoning, but they can be partially ordered with respect to complexity and domain-specificity, such that routines later in the partial order make use of routines earlier.



**Fig. 2.** A DDS for a simple diagram composed of a point, a curve and a region. There are other objects: distinguished points such as end points, the closed curve defining the periphery of the region, etc.

As objects in DDS are created, deleted or modified, new objects might emerge, objects may change their spatial extents, and existing objects might be lost. PRs will make these determinations and DDS will be updated to reflect these changes. DDS mediates the interaction of the problem solver with  $I$ . For example, given a question such as “Is A to the left of C?,” the information in DDS is used to identify the image descriptions for A and C as arguments for the PR Leftof(X,Y). In principle, the same image  $I$  might correspond to different DDS’s depending on which subsets are organized and recognized as objects, such as in the well-known example of a figure that might be perceived as a wine glass or profiles of two faces.

<sup>1</sup> We call them perceptual routines, rather than the more common and specific visual routines, because our long-term goal is to extend the notion of the cognitive state to multiple perceptual modalities, and vision is just one modality.

#### 4. PERCEPTUAL ROUTINES (PRs)

PRs can be categorized into two classes, emergent object recognition, and property/relation extraction. The first includes domain-independent PRs that identify point, curve and region objects that are created or lost when a configuration of diagrammatic objects is specified or modified. The PRs of the second class produce symbolic descriptions belonging to one of three kinds: (i) specified properties of specified objects (e.g., curve C has length of m units), (ii) relations between objects (e.g., point P is in region R, curve C<sub>1</sub> is a segment of curve C<sub>2</sub>, object O<sub>1</sub> is to the left of object O<sub>2</sub>, values of the angles made by intersection of curves C<sub>1</sub> and C<sub>2</sub>), and (iii) symbols that name an object or a configuration of objects as an instance of a class, such as a triangle or a telephone.

The PRs of the second class come in different degrees of domain specificity. Properties such as length of curve, area of a region, and quantitative and qualitative (right, acute, obtuse, etc.) values of angles made by intersections of curves are very general, as are *subsumption relations* between objects, such as that curve C<sub>1</sub> is a segment of curve C<sub>2</sub>. Relations such as *Insideof*(A, B), *Touches*(A, B), and *Leftof*(A, B) are also quite general. PRs that recognize that a curve is a straight line, a closed curve is a triangle, etc., are useful for reasoning in Euclidean geometry, along with relations such as *Parallel*(Line<sub>1</sub>, Line<sub>2</sub>). The PRs of the second class are open-ended in the sense that increasingly domain-specific perceptions may be conceived: e.g., an L-shaped region, *Half-way-between*(Point A, Point B). Our goal for the current set of PRs is what appears to be a useful general set, with the option for additional special purpose routines later on. The following is a list of PRs of different types that we have currently identified and implemented as being generally useful.

**Emergent Object Recognition Routines.** These PRs return one or more objects after creating or recognizing them. Examples of such routines are finding intersection-points when curve and/or region objects intersect, region when a curve closes on itself, new regions when regions intersect, new regions when a curve intersects with a region, extracting distinguished points on a curve (such as end points) or in a region, extracting distinguished segments of a curve (such as those created when two curves intersect), extracting periphery of a region as a closed curve. Reverse operations are included – such as when a curve is removed, certain region objects will no longer exist and need to be removed.

We have implemented the intersection routine using sweep-line algorithm [Bentley & Ottmann, 1979] that takes as input a number of curve and region objects and computes all the intersection points in  $O((n+k)*\log(n))$  time (see *Intersect* in Table 1). Due to an

intersection, a number of emergent objects are created. For e.g., when two curves intersect, four sub-curves and an intersection point emerge. However, computing all the emergent curves and regions is computationally too expensive, so we restrict ourselves to computing only the emergent first order objects – i.e., objects that do not have other emergent objects of the same type as subparts, or objects specifically requested by the problem solver.

**Object Property Extraction Routines.** These routines might return a numerical value as in *Length*(Curve C), or a boolean as in case of *Closed*(Curve C). *Length*(Curve C) computes the length of C by computing the sum of Euclidean distances between each consecutive pair of points. *Area*(Region R) computes the signed area of R, which is assumed to be an arbitrary non-self-intersecting polygon with n vertices, using the formula:

$$Area = \frac{1}{2} \sum_{i=1}^n (x_i y_{(i \bmod n)+1} - x_{(i \bmod n)+1} y_i)$$

where  $(x_i, y_i)$  is the coordinate of the  $i^{th}$  vertex. *Counter-Clock-Wise*(Region R) checks whether R is oriented in counterclockwise direction by checking whether the area of R is positive. *Angle*(Point P<sub>1</sub>, Point P<sub>2</sub>, Point P<sub>3</sub>) computes the angle between the line segments P<sub>1</sub>P<sub>2</sub> and P<sub>2</sub>P<sub>3</sub> at point P<sub>2</sub>. *Straightline*(Curve C) checks whether all the points on C are collinear or not. *Closed*(Curve C) checks whether C intersects itself by using the *Intersect* routine. Additional property extraction routines can be added as needed.

**Relational Perception Routines.** These routines return a boolean value after checking whether one or more objects satisfy a certain relation. *Insideof*(Point P, Region R) checks whether P lies inside R or not. In order to compute whether a given object of any type lies within a given region or not, we check for every point of that object using the routine *Insideof*. *Outsideof* is implemented similarly.

*Leftof*(Point P<sub>1</sub>, Point P<sub>2</sub>, POV) checks whether P<sub>1</sub> is to the left of P<sub>2</sub> or not with respect to the given point of view POV. The point of view specifies the direction towards which the observer is faced, in terms of an angle with respect to a fixed horizontal axis. *Rightof*(Point P<sub>1</sub>, Point P<sub>2</sub>, POV), *Above*(Point P<sub>1</sub>, Point P<sub>2</sub>, POV), *Below*(Point P<sub>1</sub>, Point P<sub>2</sub>, POV) can be computed similarly. *Topof*(Region R<sub>1</sub>, Region R<sub>2</sub>) is required in domains like the Blocks World where one block might be on top of another block. In such cases, we would consider two blocks as regions R<sub>1</sub>, R<sub>2</sub>, and infer that R<sub>1</sub> is on top of R<sub>2</sub> if R<sub>1</sub> and R<sub>2</sub> touch each other at more than one point and R<sub>1</sub> is above R<sub>2</sub> with respect to the vertical point of view.

**Table 1.** Selected Perceptual Routines and Their Computational Complexities

Class	PR	Input	Output	Computational complexity	PRs used
Quantitative PRs	Distance	Point $P_1$ , Point $P_2$	A real number	$O(1)$	-
	Angle	Point $P_1$ , Point $P_2$ , Point $P_3$	A real number	$O(1)$	-
	Area	Region R	A real number	$O(n)$ $n = \# \text{ segments in R}$	-
	Length	Curve C	A real number	$O(n)$ $n = \# \text{ segments in C}$	Distance
Qualitative PRs	StraightLine	Curve C	A boolean	$O(n)$ $n = \# \text{ segments in C}$	-
	Closed	Curve C	A boolean	$O((n+k)*\log(n))$ $n = \# \text{ segments in C}, k = \# \text{ intersections}$	Intersect
	Counter-Clock-Wise	Region R	A boolean	$O(n)$ $n = \# \text{ segments in R}$	Area
	Leftof	Point $P_1$ , Point $P_2$ , Point of View	A boolean	$O(1)$	-
	Rightof	Point $P_1$ , Point $P_2$ , Point of View	A boolean	$O(1)$	Leftof
	Above	Point $P_1$ , Point $P_2$ , Point of View	A boolean	$O(1)$	-
	Below	Point $P_1$ , Point $P_2$ , Point of View	A boolean	$O(1)$	Above
	On	Point P, Curve C	A boolean	$O(n)$ $n = \# \text{ segments in C}$	-
	Touches	Object $O_1$ , Object $O_2$	A boolean	$O(n_1*n_2)$ $n_1 = \# \text{ segments in } O_1, n_2 = \# \text{ segments in } O_2$	On
	Topof	Region $R_1$ , Region $R_2$	A boolean	$O(n_1*n_2)$ $n_1 = \# \text{ segments in } R_1, n_2 = \# \text{ segments in } R_2$	Touches, Above
	Insideof	Point P, Region R	A boolean	$O(n)$ $n = \# \text{ segments in R}$	-
	Outsideof	Point P, Region R	A boolean	$O(n)$ $n = \# \text{ segments in R}$	Insideof
	Subcurveof	Curve $C_1$ , Curve $C_2$	A boolean	$O(n_1*n_2)$ $n_1 = \# \text{ segments in } C_1, n_2 = \# \text{ segments in } C_2$	On
	Subregionof	Region $R_1$ , Region $R_2$	A boolean	$O(n_1*n_2)$ $n_1 = \# \text{ segments in } R_1, n_2 = \# \text{ segments in } R_2$	Insideof, Intersect
	Parallel	Curve $C_1$ , Curve $C_2$	A boolean	$O(n)$ $n = \# \text{ points in } C_1 \text{ or } C_2$	-

Table 1 continued.

Class	PR	Input	Output	Computational complexity	PRs used
Object Recognition PRs	ScanPath	Curve C, Diagram D, Relation S	Object O <sub>1</sub> , Object O <sub>2</sub> , ... Object O <sub>r</sub>	$O(n*m)$ $n = \# \text{ segments in } C, m = \# \text{ objects in } D$	PR for extracting relation S
	Intersect	Curve C <sub>1</sub> , Curve C <sub>2</sub> , ... Curve C <sub>r</sub> , Region R <sub>1</sub> , Region R <sub>2</sub> , ... Region R <sub>t</sub>	Point P <sub>1</sub> , ... Point P <sub>s</sub> , Curve C <sub>1</sub> , ... Curve C <sub>p</sub> , Region R <sub>1</sub> , ... Region R <sub>q</sub>	$O((n+k)*\log(n))$ $n = \sum_{i=1}^r \# \text{ segments in } C_i + \sum_{j=1}^t \# \text{ segments in } R_j$ $k = \# \text{ intersections}$	-

**Table 2.** Selected Action Routines and Their Computational Complexities

AR	Input	Output	Computational complexity	PRs/ARs used
Translate	Object O, Real number t	Object O'	$O(n)$ $n = \# \text{ points in the input object}$	-
Rotate	Object O, Real numbers (x, y, $\theta$ )	Object O'	$O(n)$ $n = \# \text{ points in the input object}$	-
Medial	Polygon, Polygon Vertices	Medial Axis	$O(n*\log(n))$ $n = \# \text{ sampled points in the input polygons}$	-
PathFinder	Polygon, Polygon Vertices, Start Pt, End Pt	$O(2^k)$ Paths, Path Lengths	$O(n^2+k^k)$ $n = \# \text{ sampled points in the input polygons}$ $k = \# \text{ polygons } (k \ll n)$	Medial
Homotopic	Path <sub>1</sub> , Path <sub>2</sub> , Point P <sub>1</sub> , Point P <sub>2</sub> , ... Point P <sub>n</sub>	A boolean	$O(n*p)$ $n = \# \text{ input points}$ $p = \sum_{i=1}^2 \# \text{ segments in Path}_i$	Insideof
ModifyPath_ to_avoid_ obstacles	Path, Diagram OldD, Diagram NewD	$O(2^k)$ Paths, Path Lengths	$O(n^2+k^k)$ $n = \# \text{ sampled points in the input polygons}$ $k = \# \text{ polygons } (k \ll n)$	PathFinder, Homotopic
ModifyPath_ to_pass_ through_ given_points	Path, Diagram D, Point P <sub>1</sub> , Point P <sub>2</sub> , ... Point P <sub>r</sub>	$O(2^k)$ Paths, Path Lengths	$O(r*(n^2+k^k))$ $n = \# \text{ sampled points in the input polygons}$ $k = \# \text{ polygons } (k \ll n)$ $r = \# \text{ points to pass through}$	PathFinder, Homotopic
ShortenPath	Path, Point P <sub>1</sub> , Point P <sub>2</sub> , ... Point P <sub>n</sub>	ShorterPath	$O(n*p*r)$ $n = \# \text{ input points}$ $p = \# \text{ segments in the input path}$ $r = \# \text{ iterations desired/required for convergence}$	-

`On(Point P, Curve C)` checks whether P lies on C or not. It is noteworthy that the point might not necessarily be one of the points describing the curve but still lie on the curve by lying on one of its segments. `On(Curve C1, Curve C2)` is computed by checking whether each segment of C<sub>1</sub> lies on C<sub>2</sub> or not. `Subcurveof(Curve C1, Curve C2)` checks whether C<sub>1</sub> is a sub-curve of C<sub>2</sub> or not. For C<sub>1</sub> to be a sub-curve of C<sub>2</sub>, each segment of C<sub>1</sub> must lie on C<sub>2</sub> and C<sub>1</sub> has to be smaller in length than C<sub>2</sub>. So we check whether there exists a segment of C<sub>1</sub> that does not lie on C<sub>2</sub> using PR `On` to infer the result.

`Subregionof(Region R1, Region R2)` computes whether R<sub>1</sub> is a sub-region of R<sub>2</sub> or not. In order for R<sub>1</sub> to be a sub-region of R<sub>2</sub>, R<sub>1</sub> has to be inside R<sub>2</sub>. If some points or segments belonging to the periphery of R<sub>1</sub> lie on the periphery of R<sub>2</sub>, still we consider R<sub>1</sub> to be a sub-region of R<sub>2</sub>. `Touches(Object O1, Object O2)` checks whether objects O<sub>1</sub>, O<sub>2</sub> touch each other or not. We consider O<sub>1</sub> and O<sub>2</sub> to touch each other if they have at least one point in common on the periphery but no point of one object lies inside the other object. Subsumption relations are especially important and useful to keep track of as objects emerge or vanish.

**Abstractions of groups of objects into higher level objects.** Objects may be clustered hierarchically into groups, such that different object abstractions emerge at different levels. For example, some events in military and meteorology domains, are characterized by a large number of individual moving elements, either in pursuit of an organized activity (as in military operations) in groups at different levels of abstraction, or subject to underlying physical forces (as in weather phenomena). Visualizing and reasoning about happenings in such domains are often facilitated by abstracting the mass of data into diagrams of group motions, and overlaying them on diagrams that abstract static features, like terrain, into regions and curves. Constructing such diagrams of motions at multiple levels of abstraction calls for generating multiple hierarchical grouping hypotheses at each sampled time instant, then choosing the best grouping hypothesis consistent across time instants, and hence following the groups to produce spatial representations of the spatiotemporal motions. For a detailed discussion and implementation of such a high level PR, the reader is referred to [Banerjee, et al, 2003; Chandrasekaran, et al, 2002]. Other PRs in this class might include generating associations between objects based on properties like those of Gestalt principles.

*Domain-specificity.* Perceptions may be domain-specific because they are of interest only in some domains, e.g., “an L-shaped region.” They may also be domain-specific in that they combine pure spatial perception with domain-specific, but non-spatial, knowledge. For example, in a military application, a

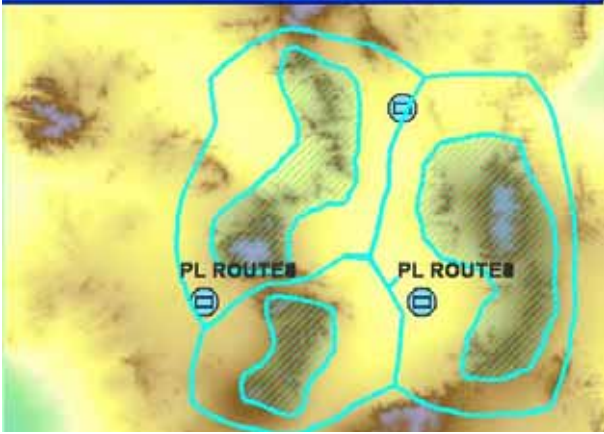
curve representing the motion of a unit towards a region might be interpreted as an attack, but that interpretation involves combining domain-independent spatial perceptions – such as extending the line of motion and noting that it intersects with the region – with non-spatial domain knowledge – such as that the curve represents the motion of a military unit, that the region’s identity is as a military target belonging to a side that is the enemy of the unit that is moving, etc. In our current implementation, it is the task of the problem solver to combine appropriately the domain-independent perceptions with domain-specific knowledge to arrive at such conclusions, but in application-dependent implementations of the architecture, some of these perceptions might be added to the set of PRs.

## 5. ACTION ROUTINES (ARs)

The problem solving process may modify the diagram – create, destroy, or modify objects. Typically, the task – the reverse of perception in some sense – involves creating the diagram such that the shapes of the objects in it satisfy a symbolically stated constraint, such as “add a curve from point A to point B that goes midway between regions R<sub>1</sub> and R<sub>2</sub>,” and “modify the object O<sub>1</sub> such that point P in O<sub>1</sub> touches point Q in object O<sub>2</sub>.” Again similar to PRs, ARs can vary in generality. Deleting named objects that exist in the diagram, and adding objects with given spatial specifications, e.g., Add point at coordinate, Add curve <equation>, etc., are quite straightforward. Our ARs include translation and rotation of named objects for specified translation and rotation parameters.

The military domain calls for a special type of action routine that constructs paths satisfying constraints. The entity-reidentification example calls for ARs that find one or more *representative paths* from point A to point B, such that intersections with a given set of objects are avoided. A representative path is a curve object and has the right qualitative properties (e.g. avoid specific regions), but is a representative of a class of paths with those qualitative properties, members of which may differ in various quantitative dimensions, such as length. Given a set of regions in a boundary of interest, the medial axis of the boundary considering the regions as holes in it can be computed in  $O(n \cdot \log(n))$  time where n is the total number of sampled points in the boundary and the regions [Kirkpatrick, 1979]. It can be easily shown that at least one path from any homotopy class can be derived from the medial axis. Since the number of homotopy classes is infinite, we extract only those paths from the medial axis that do not intersect themselves, and consider them as representative paths (see Fig 3). There will be at most  $O(2^k)$  representative paths where k is the number of regions. Our `Medial` and `PathFinder` ARs compute the medial axis and the representative paths respectively.

For their computational complexities, see Table 2. Two paths are considered homotopic if one can be continuously deformed into the other without crossing any obstacle. The AR Homotopic computes whether two given paths, with the same endpoints, are homotopic to each other or not by checking whether there exists any point inside the region(s) formed by the paths.



**Fig 3.** Path generation using AR Medial.

ARs can also modify given paths to satisfy certain constraints, for e.g., a given path might need to be modified because of detection of a new obstacle (region object) along its way, or a given path might be required to pass through certain given points, and so on. We have implemented routines that perform certain tasks useful for the information fusion domain. The AR `ModifyPath_to_avoid_obstacles` modifies a path to avoid newly found obstacles, by extracting all the representative paths from the new set of obstacles (old set of obstacles and the newly found obstacles) and outputs those representative paths which are homotopic to the given path with respect to the old set of obstacles. The AR `ModifyPath_to_pass_through_given_points` modifies a path to pass through a sequence of points, by considering each consecutive pair of points in the sequence as the starting point and the end point and extracting all representative paths between them, and then concatenating the paths to end up with at most  $O(r \cdot 2^k)$  paths, where  $k$  is the number of obstacles and  $r$  is the number of points in the sequence. The AR outputs only those paths that are homotopic to the given path. Table 2 gives the computational complexities of these routines. It is noteworthy that these ARs are not primitive ARs, rather they are built using primitive ARs such as `Medial` and `PathFinder`.

The set of ARs also includes routines that adjust a path in a homotopy class to be shorter, longer, etc., in various ways. One useful AR is shortening a given path. Finding the shortest path in a given homotopy class is a very well-defined problem in computational geometry. In

our case, we are interested not always in the shortest path but also in the shorter versions of a path in a given homotopy class as that saves computational costs in many cases while in some other cases, we just need a smooth path that is close to being shortest but not an absolute shortest as the shortest path might have sharp turns through which it is sometimes difficult to navigate. The AR `ShortenPath` shortens a path gradually until the absolute shortest configuration is reached. After each iteration, the algorithm produces a path shorter than the one after the last iteration.

Extending lines indefinitely in certain directions so that a PR can decide if the extended line will intersect with an object of interest is one that we need in our domain. Other researchers have found specific sets of ARs that are useful for their tasks, such as the AR in [Lindsay, 1998], “Make a circle object that passes through points A, B, and C.” An AR that we have not yet used, but we think would be especially valuable, is one that changes a region object into a point object and vice versa as the resolution level changes in problem solving, such as a city appearing as a point in a national map, while it appears as a region in a state map.

*Underspecification of spatial properties of objects.* More generally, each of the PRs can be reversed and a corresponding AR imagined. For example, corresponding to the PR `Insideof( $R_1, R_2$ )` is the AR, “Make region  $R_2$  such that `Insideof( $R_1, R_2$ )` is true,” (assuming region  $R_1$  exists); and corresponding to `Length(curve  $C_1$ )` is the AR, “Make curve  $C_1$  such that `Length( $C_1$ )` < 5 units.” In most such instances, the spatial specification of the object being created or modified is radically under-defined. Depending on the situation, random choices may be made, or certain rules about creation of objects can be followed. However, the problem solver needs to keep track of the fact that the reasoning system is not committed to all the spatial specification details.

## 6. RELATED WORK

Ullman [Ullman, 1984] proposed, there exists a fixed set of low-level elemental operations, such as shifting of processing focus, selection of salient locations, defining a region of interest, marking locations already visited, etc. that might be efficiently composed into visual routines, such as visual search, texture segregation, contour grouping, and in this manner extract an essentially unbounded variety of shape properties and spatial relations. A closely related procedural approach was proposed in [Just & Carpenter, 1976], examining visual tasks such as mental rotation, from a higher-level perspective.

Hayhoe [Hayhoe, 2000] argues that vision can be thought of as the ongoing execution of task-specific

routines which can be composed into extended behavioral sequences. Based on Newell's conceptualization of brain's temporal hierarchy [Newell, 1990], Hayhoe showed using an example of autonomous driving that the routines depend critically on the immediate behavioral context. For the same domain, visual routines such as traffic light detection, stop sign detection, intersection detection, looming detection, vehicles detection, obstacle detection, road detection, etc. were developed [Salgian & Ballard, 1998]. Rao and Ballard [Rao & Ballard, 1995] proposed visual routines such as object identification, object location identification, looming detection -- composing these with different parameters allows complex visual behaviors to be obtained.

Spatial relations have been classified in different classes -- topological relations (e.g. disjoint), direction relations (e.g. north, east), distance relations (e.g. far, near), inclusion relations (e.g. in, at), and fuzzy relations (e.g. next to, close to) [Pullar & Egenhofer, 1988].

Our notion of PRs is based on a notion of composable and extensible primitives, but more oriented to the needs of problem solving with diagrams. These routines operate on interpreted images i.e. in the realm of what is referred to as transformation (imagery) processes in [Papadias & Kavouras, 1994]. Because of our interest in generic objects, aspects of our proposals are intended to be domain-independent as much as possible.

## 7. CONCLUSIONS

An architecture for representing and reasoning with diagrams, and its applications to some situation understanding and planning problems of Army interest have been described in our earlier work. The focus of the current paper has been on the family of perceptual and diagram construction algorithms -- or, perception and action routines as we have called them -- that have been found useful in these applications. We describe their algorithmic basis, and characterize their computational complexity properties. Research of the type reported will help in building practical decision support systems with manageable complexity. Even though they were motivated by Army problems, we believe that the routines described are applicable to diagrammatic reasoning in general. While the set of such routines is open-ended, we think that the routines we have described will provide a good portion of the base set out of which more complex routines can be built.

## ACKNOWLEDGMENTS

This research was supported by participation in the Advanced Decision Architectures Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory under Cooperative Agreement

DAAD19-01-2-0009. We acknowledge the help of John Josephson, Unmesh Kurup, Vivek Bharathan, and Robert Winkler for collaboration that helped in clarifying the ideas in the paper.

## REFERENCES

- Banerjee, B., Chandrasekaran, B., Josephson, J.R., Winkler, R., 2003: "Constructing Diagrams to Support Situation Understanding and Planning: Part 1: Diagramming Group Motions," *Technical Report OSU-CISRC-5/03-TR29*, Dept. of Computer Sc. & Engg., Ohio State University, Columbus.
- Bentley, J., Ottmann, T., 1979: "Algorithms for reporting and counting geometric intersections," *IEEE Transactions on Computers*, C-28:643-647.
- Chandrasekaran, B., Josephson, J.R., Banerjee, B., Kurup, U., Winkler, R., 2002: "Diagrammatic Reasoning in Support of Situation Understanding and Planning", *Proc. 23<sup>rd</sup> Army Science Conference*, FL.
- Chandrasekaran, B., Kurup, U., Banerjee, B., Josephson, J.R., Winkler, R., 2004: "An Architecture for Problem Solving with Diagrams," appears in *Diagrammatic Representation and Inference*, Alan Blackwell, Kim Marriott, Atsushi Shimojima, Editors, Lecture Notes in AI 2980, Berlin: Springer-Verlag, 151-165.
- Hayhoe, M., 2000: "Vision using routines: A functional account of vision", *Visual Cognition*, 7(1-3):43-64.
- Josephson, J.J., Josephson, S.J., 1996: *Abductive Inference: Computation, Philosophy, Technology*, Cambridge University Press.
- Just, M., Carpenter, P., 1976: "Eye fixations and cognitive processes," *Cognitive Psychology*, 8:441-480.
- Kirkpatrick, D., 1979: "Efficient computation of continuous skeletons," *Proc. 20<sup>th</sup> IEEE Symposium on Foundations of Computing*, 28-35.
- Lindsay, R.K., 1998: "Using diagrams to understand geometry," *Computational Intelligence*, 14(2):238-272.
- Newell, A., 1990: *Unified theories of cognition*. Harvard University Press.
- Papadias, D., Kavouras, M., 1994: "Acquiring, representing and processing spatial relations," *Proc. 6<sup>th</sup> Intl. Symposium on Spatial Data Handling*, Edinburgh, U.K., Taylor Francis.
- Pullar, D., Egenhofer, M., 1988: "Towards formal definitions of topological relations among spatial objects," *Proc. 3<sup>rd</sup> Intl. Symposium on Spatial Data Handling*.
- Rao, R.P.N., Ballard, D.H., 1995: "An active vision architecture based on iconic representations," *Artificial Intelligence Journal*, 78:461-505.
- Salgian, G., Ballard, D.H., 1998: "Visual routines for autonomous driving," *Proc. 6<sup>th</sup> Intl. Conf. on Computer Vision*, Bombay, India, 876-882.
- Ullman, S., 1984: "Visual Routines," *Cognition*, 18:97-159.